Universalities in cellular automata; a (short) survey

Nicolas Ollinger

Journées Automates Cellulaires 2008 (Uzès), pp. 102-118

UNIVERSALITIES IN CELLULAR AUTOMATA A (SHORT) SURVEY

NICOLAS OLLINGER [1]

[1] Laboratoire d'informatique fondamentale de Marseille (LIF)
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France
E-mail address: Nicolas.Ollinger@lif.univ-mrs.fr

Abstract. This reading guide aims to provide the reader with an easy access to the study of universality in the field of cellular automata. To fulfill this goal, the approach taken here is organized in three parts: a detailed chronology of seminal papers, a discussion of the definition and main properties of universal cellular automata, and a broad bibliography.

Introduction

The idea and construction of a universal cellular automaton is as old as the formal study of the object itself, starting with the work of von Neumann [82] on self-reproduction in the 1940s, using cellular automata under suggestions by Ulam. Following the work of Turing, a Turing-universal cellular automaton is an automaton encompassing the whole computational power of the class of Turing machines, or by so-called Church-Turing thesis the class of recursive functions. To encode complex behaviors in a cellular automaton's dynamics, one can describe how to encode any computing device of a universal class of machines (Turing machine, tag systems, etc) and use classical tools of computability theory to shape wanted behaviors of the object. This is basically what von Neumann did. He designed a cellular automaton able to encode any Turing machine, the machine being moreover equipped with a construction arm controlled by the machine's head.

But Turing-universality is not the only reasonable kind of universality one might expect from cellular automata. It is quite unnatural to consider a universality of highly parallel potentially infinite devices as cellular automata by simulation of the dynamics of sequential finite machines — indeed, as we will discuss, to give a both widely acceptable yet precise definition of Turing-universality is a very difficult and unfulfilled challenge. As the study of cellular automata shifted both to dimension 1 and to the study of its dynamics, another kind of universality emerged. An intrinsically universal cellular automaton is an automaton able to properly simulate the behavior of any other cellular automaton on any type of

2000 ACM Subject Classification: F.1.1.
Key words and phrases: cellular automata, universality, reversibility.
This research has been supported by the French national research agency (ANR) grant Sycomore.

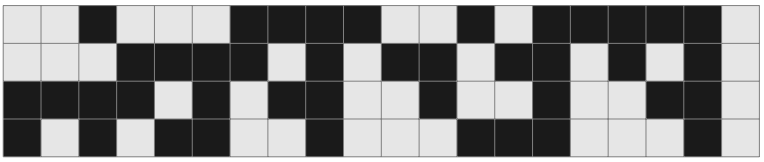

Figure 1: Partial space-time diagram of the $(\mathbb{Z}_2, +)$ rule

configuration (might it be infinite). It turns out that most of the historical constructions in dimension 2 and more, whereas designed as Turing-universal are Intrinsically Universal by the simple fact that they are designed to encode any boolean circuit.

A formal definition of universality might not seem so important. In fact, when building a precise cellular automaton from scratch to be universal, a definition is often implicit: the obtained behavior is the one engineered by the designer. The definition turns out to be more required when proceeding by analysis: given a cellular automaton rule, is it universal?

The present reading guide is constructed as follows. Section 1 *Cellular Automata* (p. 103) gives the definitions and notations used for cellular automata, configurations, and dynamics. Section 2 *Chronology* (p. 105) is an annotated chronology of seminal papers preparing to and concerning universality and universal cellular automata. Section 3 *Towards Formal Definitions* (p. 108) discusses the right definition of universalities in cellular automata. Section 4 *Higher Dimensions* (p. 110) discusses the construction and analysis of universal cellular automata in dimensions 2 and more, mostly using boolean circuits simulation. Section 5 *Turing Universality* (p. 111) discusses Turing universality, its links with universal Turing machines and the main technics of construction. Section 6 *Intrinsic Universality* (p. 113) discusses Intrinsic universality and the main technics of construction. Section 7 *Reversiblity and Universality* (p. 115) discusses universality in the special restricted case of reversible cellular automata.

1. Cellular Automata

A *cellular automaton* $\mathcal{A}$ is a tuple (d, S, N, f) where d is the *dimension* of space, S is a finite set of *states*, N a finite subset of $\mathbb{Z}^d$ is the *neighborhood* and $f : S^N \to S$ is the *local rule*, or *transition function*, of the automaton. A *configuration* of a cellular automaton is a coloring of the space by S, an element of $S^{\mathbb{Z}^d}$. The *global rule* $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ of a cellular automaton maps a configuration $c \in S^{\mathbb{Z}^d}$ to the configuration $G(c)$ obtained by applying f uniformly in each cell: for all position $z \in \mathbb{Z}^d$, $G(c)(z) = f(c(z + \nu_1), \ldots, c(z + \nu_k))$ where $N = \{\nu_1, \ldots, \nu_k\}$. A *space-time diagram* of a given cellular automaton is a mapping $\Delta \in S^{\mathbb{N}\times\mathbb{Z}^d}$ such that for all time step $t \in \mathbb{N}$, $\Delta(t + 1) = G(\Delta(t))$.

Example 1.1. Fig. 1 is a partial representation of a space-time diagram of the cellular automaton $(1, \mathbb{Z}_2, \{0, 1\}, f)$ where $f(x, y) = x + y$. State 0 is represented by the white color, state 1 by the black color. Time goes from bottom to top.

In this paper, we consider for the most part cellular automata of dimension 1 and 2 with the typical neighborhoods depicted on Fig. 2: von Neumann $\{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ and Moore $\{-1, 0, 1\}^2$ in dimension 2, first neighbors $\{-1, 0, 1\}$ and one way $\{-1, 0\}$ in dimension 1.

(a) von Neumann (b) Moore (c) first neighbors (d) one way

Figure 2: Typical neighborhoods

Several subsets of the space of configurations are considered. Given a *quiescent state* q satisfying $f(q,\dots,q) = q$, a *q-finite configuration* c is a configuration equal to q in all but finitely many cells: there exists α such that for all position $z \in \mathbb{Z}^d$, $\|z\|_\infty > \alpha \to c(z) = q$. A configuration c admits p as a *periodicity vector* if for all position $z \in \mathbb{Z}^d$, $c(z+p) = c(z)$. A configuration c in dimension d is *periodic* if it admits a family of d non-colinear periodicity vectors: there exists $p \in \mathbb{N}^d$ such that $(p_1, 0, \dots, 0)$, $(0, p_2, 0, \dots, 0)$, $\dots$, and $(0, \dots, 0, p_d)$ are periodicity vectors of c. A configuration c in dimension d is *ultimately periodic* if there exists α and d non-colinear vectors v_i such that for all position $z \in \mathbb{Z}^d$ and all vector v_i, $\|z\|_\infty > \alpha \to c(z+v_i) = c(z)$. Notice that in dimension 1, an ultimately periodic configuration can have two different ultimately periodic pattern on each side.

Constraints can also be added to the local rule. Symmetries are usually considered to obtain more natural rules mimicking physical systems. A symmetry rule can be seen as a one-to-one mapping $\rho : \mathbb{Z}^d \to \mathbb{Z}^d$: the image of a configuration $c \in S^{\mathbb{Z}^d}$ by the symmetry rule ρ is the configuration $\rho(c)$ satisfying for all position $z \in \mathbb{Z}^d$, $\rho(c)(z) = c(\rho(z))$. A cellular automaton $\mathcal{A}$ respects a symmetry rule ρ if ρ and G commute, *i.e.* $\rho(G(c)) = G(\rho(c))$. Typical symmetries include reflections around point ($\rho_0(x,y) = (-x,-y)$), around axes ($\rho_x(x,y) = (-x,y)$) and rotations ($\theta(x,y) = (-y,x)$). A cellular automaton is *totalistic* if it's set of states is a subset of $\mathbb{N}$ and the local rule f can be written as $f(s_1,\dots,s_k) = g(\sum_{i=1}^k s_i)$. Totalistic rules respect all symmetries that preserve the neighborhood (*i.e.* such that the image of the neighborhood by the symmetry rule is equal to the neighborhood): totalistic cellular automata with the von Neumann or Moore neighborhood are reflection and rotation invariants.

A cellular automaton is injective (resp. surjective, one-to-one) if its global rule is injective (resp. surjective, one-to-one). A cellular automaton $\mathcal{A}$ is reversible if there exists a cellular automaton $\mathcal{B}$ that reverts it, that is such that $G_\mathcal{B} \circ G_\mathcal{A}$ is the identity map.

Proposition 1.2 (Hedlund [31], Richardson [72])**.** *A cellular automaton is reversible if and only if it is injective.*

Proposition 1.3 (Amoroso and Patt [2])**.** *It is decidable given a one-dimensional cellular automaton to decide whether it is reversible.*

Proposition 1.4 (Kari [34, 35])**.** *It is undecidable given a two-dimensional cellular automaton to decide whether it is reversible.*

Whereas reversibility is an undecidable question, the construction of reversible cellular automata is possible, provided that the backward rule is constructed at the same time as the forward rule. Partitioned cellular automata provide a convenient way to construct reversible cellular automata. A *partitioned cellular automaton* is a cellular automaton with state set

$S_1 \times S_2 \times \cdots \times S_k$ whose local rule can be rewritten as $f((s_1^1, \ldots, s_1^k), \ldots, (s_k^1, \ldots, s_k^k)) = \varphi(s_1^1, s_2^2, \ldots, s_k^k)$ where $\varphi : \prod S_i \to \prod S_i$ is the partitioned rule. As it is straightforward to verify, a partitioned cellular automaton is reversible if and only if its partitioned rule is one-to-one. As the partitioned rule is a mapping from a finite set to itself, any partially defined injective rule can be completed to a reversible cellular automaton.

For a better and more complete introduction to the theory of cellular automata, see Delorme [18] and/or Kari [36].

2. Chronology

It is a difficult task to give a fair and complete chronology of a research topic. In this section, we propose an exploration of the history of the field in three main eras:

(1) the *computation and machines* era describes seminal papers outside the realm of cellular automata that lead to the main tools necessary to consider computation in the context of abstract machines;
(2) the *universality and cellular automata* era is the core part of the chronology: it describes seminal papers along the path of universality study in the realm of cellular automata, from the early work of von Neumann in the 50s to the end of the 90s;
(3) the *recent trends* era is a more subjective choice of some papers in the field in the twenty-first century.

2.1. Computation and Machines

Gödel 1931 [30]**:** in his now classical paper describing incompleteness theorems, Gödel introduces so-called Gödel numberings: the ability to encode and manipulate a formal system inside itself if the system is complex enough. The concept of universality directly depends on such an encoding: a universal machine simulates a machine through its encoding. For a precise analysis from a logic and computer science point of view of Gödel's paper, see Lafitte [38].

Turing 1936 [81]**:** while introducing Turing machines and proving the undecidability of the halting problem by a diagonal argument, Turing also introduces its universal machine. Fixing an enumeration of Turing machines and a recursive bijective pairing function $\langle ., . \rangle : \mathbb{N}^2 \to \mathbb{N}$, he describes a machine U that, on input $\langle m, n \rangle$, computes the same value as the machine encoded m on input n. Universality as a property is not discussed: a unique universal Turing machine U is given. For a discussion of the development of ideas from Leibniz to Turing results, see Davis [17].

Post 1943 [68]**:** at that time, many different models of computation where proposed and proved equivalent, leading to the so-called Church-Turing thesis. Post introduces tag systems, a combinatorial word-based system successfully used since to construct size-efficient universal Turing machines. For a modern definition and discussion of tag systems, see Minsky [51].

Kleene 1956 [37]**:** finite state machines are at the hearth of many models of computation. Kleene's paper proves the equivalence between three different families of objects: regular languages, finite automata, and boolean circuits. Boolean circuits are modeled after the formal study of abstract neurons by McCulloch and Pitts [49]. This equivalence is fundamental both to concrete computer design and discrete models of computation like cellular automata. For a modern discussion on

this equivalence and its consequences from the point of view of computation and machines, see Minsky [51]. Perrin [66] gives an history of this period and important achievements with respect to the field of finite automata and formal languages.

Minsky 1967 [51]**:** In the spirit of the question from Shannon [73] about the size of a smallest Turing machine, Minsky explains how to efficiently encode tag systems computations into Turing machines and describe a universal Turing machine with four symbols and seven states. This marks the real start of a (still running) competition.

Lecerf 1963 [40]**, Bennett 1973** [8]**:** Reversible computation is concerned with computing devices that can unroll their computation, going back in time. In their independent papers, Lecerf and Bennett prove that reversible Turing machines are able to simulate just any Turing machine. Thus, there exists reversible Universal Turing machines.

Fredkin and Toffoli 1982 [27]**:** To encode classical computations into discrete models, Kleene [37]'s theorem permits to go freely from circuits to finite state machine, an essential ingredient for computation. Fredkin and Toffoli discuss an analogous for reversible computation: elementary building blocks to encode any reversible finite state machine as a reversible circuit. This paper also introduces the so-called billard ball model of computation: a discrete cellular automata model to encode reversible computations. The encoding of reversible finite state machines into circuits was later improved by Morita [55].

2.2. Universality and Cellular Automata

von Neumann 1966 [82]**:** Introducing cellular automata in order to construct a self-reproducing machine, participating to the reflexion on the nature of life, von Neumann takes a fixed-point approach. His two-dimensional, 29 states, von Neumann neighborhood cellular automaton is able to simulate a particular kind of Turing machine that can also control a construction arm. The power of the construction arm is rich enough to construct with finitely many instructions a copy of the Turing machine itself. Whereas the machine is constructed with a form of Turing-universality in mind, the simulation of the Turing machine is done with very simple components wiring down a particular family of boolean circuits. As a consequence, the original cellular automaton is also, *a posteriori*, intrinsically universal. The construction of von Neumann leads to various improvements and discussions on the encoding of boolean circuits, the different organs that compose the machine and the transmission of signals. A non exhaustive list of interesting following papers might be: Arbib [3], Burks [10, 11, 12], Moore [52, 53], Thatcher [77, 76].

Codd 1968 [14]**:** Following the principle of von Neumann idea on self-reproduction, Codd drastically reduces the complexity of the automaton. Codd's two-dimensional rule uses 8 states with the von Neumann neighboorhood. Signals are conveyed by pairs of states (an oriented particle) moving between walls and reacting upon collision. This cellular automaton is also universal for boolean circuits and so intrinsically universal. A latter construction by Langton [39], based on Codd ideas, has fewer states and a very simple family of self-reproducing loops but looses its computation universal capabilities.

Banks 1970 [4, 5]**:** The work of Banks is noticeable with respect to several aspects and also because of its relatively small diffusion in the cellular automata community. Banks constructs a family of very small cellular automata (two-dimensional, von Neumann neighborhood, very symmetric, four to two states) simulating boolean circuits in a very simple and modern way (signals moving in wires, boolean gates on collisions), he identified and used explicitly the property of intrinsic universality and gave a transformation to construct relatively small universal one-dimensional cellular automata with large neighborhoods starting from two-dimensional ones (reencoding it into a one-dimensional first-neighbors automaton with 18 states). Construction of a two-dimensional four state universal cellular automaton in the spirit of Banks is provided by Noural and Kashef [61].

Conway 1970 [29, 9]**:** The Game of Life introduced by Conway is certainly among the most famous cellular automata and the first rule to be proven universal by analysis of a given rule rather than on purpose construction. A modern exposition of the Game of Life universality and a proof of its intrinsic universality was latter proposed by Durand and Róka [22].

Smith III 1971 [74]**:** The simulation of Turing machine by cellular automata to construct one-dimensional Turing-universal cellular automata is studied by Smith III. Among several results, he explains how to construct a one-dimensional Turing-universal cellular automaton with first neighbors and 18 states.

Toffoli 1977 [80]**:** Any cellular automaton of dimension d can be simulated, in a certain sense, by a cellular automaton of dimension $d+1$. Using this assertion, Toffoli shows that two-dimensional reversible cellular automata can be Turing-universal. The result was later improved by Hertling [32].

Margolus 1984 [42]**:** Whereas Toffoli transforms any Turing machine into a two-dimensional cellular automaton by using a new spatial dimension to store computational choices, Margolus constructs a Turing-universal two-dimensional reversible cellular automaton by simulation a bouncing billard ball, complex enough to compute any reversible boolean function of conservative logic. The billard ball model cellular automaton has 16 states defined as two-by-two blocks of binary cells and von Neumann neighborhood.

Albert and Čulik 1987 [1]**:** Each cellular automaton can be simulated by a totalistic cellular automaton with one-way neigborhood. With the help of the last proposition, Albert and Čulik construct the first universal cellular automaton obtained by simulation of any cellular automaton of the same dimension. The automaton works along the following principle: each macro-cell copies the state of its left neighbor and adds it to its state obtaining some n, then by copying the nth element of a reference table, it selects its new state. Whereas the spirit of intrinsic universality is definitely there, the technical implementation is less clear. The one-dimensional first-neighbors automaton obtained has 14 states. The construction was later improved by Martin [43, 44] with better transition time complexity and smn theorem.

Morita and Harao 1989 [57]**:** Introducing partitioned cellular automata, Morita and Harao explicitely simulate any reversible Turing machine on a one-dimensional reversible cellular automaton, proving that one-dimensional reversible cellular automata can be Turing-universal. The construction was later improved by Dubacq [21], simulating any Turing machine in real time (without loss of time).

Lindgren and Nordahl 1990 [41]**:** The direct simulation of Turing machine on one-dimensional cellular automata proposed by Smith III can be improved and any m states n symbols machine can be simulated by a $(m + n + 2)$-states cellular automaton following Lindgren and Nordhal. Applying this to Minsky's 7 states and 4 symbols machine and then transforming the simple simulation into a macro-state signal based simulation, Lindgren and Nordhal obtain a one-dimensional first neighbors 7 state Turing-universal cellular automaton. The intrinsic universality status of this automaton is unknown.

Durand and Róka 1996 [22]**:** Revisiting the Game of Life and filling holes in the universality proof, Durand and Róka publish the first discussion on the different kinds of universality for cellular automata and the problem of formal definition.

Durand-Lose 1997 [25]**:** Using a modern definition of intrinsic universality, Durand-Lose goes one step further than Morita and Harao by constructing a one-dimensional cellular automata intrinsically simulating any cellular automaton.

2.3. Recent Trends

Imai and Morita 2000 [33]**:** The improvement in the construction of small and simple two-dimensional reversible cellular automata continues. Imai and Morita use partitioned cellular automata to define an 8 state universal automaton.

Ollinger 2002 [64]**:** Using simulation technics between cellular automata, strong intrinsically universal cellular automata with few states can be constructed, here 6 states.

Cook 2004 [15]**:** Very small universal cellular automata cannot be constructed, they have to be obtained by analysis. Realising a real tour de force, Cook was able to prove the Turing-universality of the 2-states first-neighbors so-called rule 110 by analysing signals generated by the rule and their collisions. The intrinsic universality of this automaton remains open. The original construction, simulation of a variant of tag system, was exponentially slow. For a proof of Cook's result using signals, see Richard [70].

Neary and Woods 2006 [60]**:** Recently, the prediction problem of rule 110 was proven P-complete by Neary and Woods by a careful analysis and modification of Turing machines simulation technics by tag systems. As P-completeness is required for intrinsic universality, this is another hint of the potential strong universality of rule 110.

Richard 2008 [71]**:** The limits of constructed small intrinsically universal cellular automata are converging towards analysed cellular automata. Using particles and collisions, Richard was recently able to construct a 4 state intrinsically universal first-neighbors one-dimensional cellular automaton.

3. Towards Formal Definitions

What is a universal cellular automaton? At first, the question might seem simple and superficial: a universal cellular automaton is an automaton able to compute anything recursive. In fact, a formal definition is both required and difficult to obtain. The requirement for such a definition is needed to define a frontier between cellular automata of maximal complexity and the others: in particular when considering the simplest cellular automata,

to be able to identify the most complex. The difficulty arise from the fact that we both want a definition broad enough to encapsulate all constructions of the literature and all *fair enough* future constructions. For more details concerning this philosophical question, see Durand and Róka [22] attempt to give formal definitions.

Turing-universality is the easiest form of universality one might think about, that is with a computability culture: let the cellular automaton *simulate* a well known universal model of computation, either simulating one universal object of the family or any object of the family.

The first approach pushes back the problem to the following one: what is a universal Turing machine? a universal tag system? In its original work, Turing did not define universal machines but *a* unique universal machine. The definition of universality for Turing machines was later discussed by Davis [16] who proposed to rely on recursive degrees, defining universal machines as machines with maximal recursive degree. This definition while formal lacks precise practical view of encoding problems: the issue continues to be discussed in the world of Turing machines, becoming more important as smaller and smaller universal machines are proposed. For a view on the universality of Turing machines and pointers to literature related to the topic, see Woods [86].

The second approach leads to the problem of heterogeneous simulation: classical models of computation have inputs, step function, halting condition and output. Cellular automata have no halting condition and no output. As pointed out by Durand and Róka [22], this leads to very tricky encoding problems: their own attempt of a Turing-universality based on this criterium as encoding flaw permitting counter-intuitively to consider very simple cellular automata as universal.

Turing-universality of dynamical systems in general and cellular automata in particular has been further discussed by Delvenne, Kůrka, and Blondel [19], and Sutner [75]. None of the proposed definition are completely convincing so forth, so we will choose on purpose not to provide the reader with yet another weak formal definition.

Intrinsic universality, on the other hand, is easier to formalize, yet more robust notion (in the sense that variations along the lines of the definition lead to the same set of universal automata). Consider a homogenous type of simulation: cellular automata simulated by cellular automata in a shift invariant, time invariant way. A natural type of universal object exist in this context: cellular automata able to simulate each cellular automaton. Following the ideas of grouping and bulking [69, 47, 63], we introduce a general notion of simulation broad enough to scope all reasonable constructions of the literature.

Direct simulation between two cellular automata can be formalized as follows. A cellular automaton $\mathcal{B}$ *directly simulates* a cellular automaton $\mathcal{A}$, denoted $G_{\mathcal{A}} \prec G_{\mathcal{B}}$, of the same dimension according to a mapping $\varphi : S_{\mathcal{A}} \to 2^{S_{\mathcal{B}}}$ if for any pair of states $a, b \in S_{\mathcal{A}}$, $\varphi(a) \cap \varphi(b) = \emptyset$ and for any configuration $c \in S_{\mathcal{A}}^{\mathbb{Z}^d}$, $G_{\mathcal{B}}(\varphi(c)) \subseteq \varphi(G_{\mathcal{A}}(c))$.

For any state set S, let $(m_1, \dots, m_d)$ be a tuple positive integers, the *unpacking bijective map* $o_{(m_1,\dots,m_d)} : \left(S^{\prod m_i}\right)^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is defined for any configuration $c \in \left(S^{\prod m_i}\right)^{\mathbb{Z}^d}$ and any position $z \in \mathbb{Z}^d$ and $r \in \prod_i \mathbb{Z}_{m_i}$ as $o_{(m_1,\dots,m_d)}(c)(m_1 z_1 + r_1, \dots, m_d z_d + r_d) = c(z)(r)$. The *translation* of vector $v \in \mathbb{Z}^d$ is defined for any configuration $c \in S^{\mathbb{Z}^d}$ and position $z \in \mathbb{Z}^d$ as $\sigma_v(c)(z) = c(z - v)$.

Simulation between two cellular automata is extended by considering packing, cutting and shifting of the two cellular automata such that direct simulation occur between both transformed objects. Universal objects are then maximum of the induced pre-order. In fact, it can be proved that simulation on one side is sufficient for universal objects.

Definition 3.1 (intrinsic universality)**.** A cellular automaton $\mathcal{U}$ is intrinsically universal if for each cellular automaton $\mathcal{A}$ of the same dimension there exists an unpacking map o_m, a positive integer $n \in \mathbb{N}$ and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Proposition 3.2 (Mazoyer and Rapaport [69, 47])**.** *No cellular automaton is intrinsically universal in real time (that is, when constraining cutting constant n to be equal to* $\max(m)$*): simulation cannot perform both information displacement and transition computation at the same time.*

Proposition 3.3 (Ollinger [65])**.** *Given a cellular automaton, it is undecidable to determine whether it is intrinsically universal.*

Turing-universality and intrinsic universality notions are really different notions. Some erroneous claims by Wolfram [83, 84] affirm for example that rule 110 is intrinsically universal. In fact, the question is yet open, Turing universality is the only proven thing.

Proposition 3.4 (Ollinger [63], Theyssier [78])**.** *There exists Turing-universal cellular automata which are not intrinsically universal. Moreover, some of them are at the bottom of an infinite increasing chain of equivalences classes of the preorder.*

Universality can also be discussed when considering language recognition or computation on grids. This topic is out of scope of the present paper. For more on this topic, see Mazoyer [45, 46].

4. Higher Dimensions

In two and more dimensions, an easy way to construct both intrinsically and Turing universal cellular automata is to go through boolean circuit simulation. Boolean circuits can encode any finite state machine and a cell of a cellular automaton or the control and tape of a Turing machine can be described as finite state machines. The topic of boolean circuit simulation with cellular automata is quite popular and a lot has been written on it, see for example recreations around the wireworld cellular automaton designed by Silverman and discussed in Dewdney [20]. Let us just here give technical hints and possible exotic extensions without entering details.

To simulate boolean circuits, one typically needs to mix the following ingredients:

wires: boolean signals travel in piecewise straight line in space, their paths are the wires. Several encoding of boolean signals with or without explicit wires are possible: moving particles encoded as states with a direction vector, bouncing on walls to turn as in game of life [29]; wire path encoded as wire cells with explicit direction vector on each wire cell as in von Neumann [82]; undirected wire cells on which directed signals travel; undirected wire cells on which pairs of two different signal cells travel, the direction being given by the orientation of the pair as in wireworld [20]; pairs of undirected wire paths in between which pairs of two different signal cells travel as in Codd [14] or Banks [5].

turn and delay: boolean signals should be able to turn in space and delay their arrival to permit signal synchronization.

signal crossing: in order to encode all boolean circuits, crossing of signals has to be encoded either explicitly (adding crossing states) or implicitly using delaying technics (as in von Neumann [82]) or boolean logic tricks.

gates: signals must be combined using boolean gates at least taken in a boolean universal family of gates. AND, OR, NOT is the classical one but NAND or NOR is sufficient alone.

fan-out: signals must be duplicated in some way either with an explicit fan-out state or using specific wire split rules.

Remarks and encoding tricks regarding boolean circuit simulation:

- Universal boolean functions families and their expressive power are described in Post [67]. But, in cellular automata encoding, it is easy to use constants and multiple wires encoding, thus the number of boolean classes depending on the implemented gates is finite and small.
- Clocks are only needed when dealing with some form of synchronized logic simulation. It is often used because boolean signals are encoded with two values: empty wire or signal on wire. With such an encoding, NOT gate has to generate new signal on wire and clock signal is used to determine at which time steps to do so. However, a classical coding trick to avoid the use of clocks and diods is to only implement OR and AND gates and use the two wires trick to gain boolean universality: a signal is encoded as one signal on one wire, the second being empty (thus no signal is encoded as no signal on both wires), then the NOT gate is just the wire crossing $(x, y) \mapsto (y, x)$, the AND gate can be encoded as $(x, y) \mapsto (x \wedge y, x \vee y)$ and the OR gate as $(x, y) \mapsto (x \vee y, x \wedge y)$. As both OR and AND produce signal only if there is at least one signal in input, the need for clock vanishes.
- Wire crossing can be gained for free by using the XOR gate as planar crossing can be implemented with XORs.
- Delays come for free if the wires can turn in all directions.
- In dimension 3, wire crossing is not needed, use the third dimension to route wires.
- Signal encoding can be done using signal constructions, in the spirit of Mazoyer and Terrier [48], in order to reduce the number of states.

Of course, boolean circuit simulation is not restricted to square grids. As an example of a more exotic lattice, Gajardo and Goles [28] encoded a boolean circuit simulator on a hexagonal lattice (with proper cellular automata definition).

Small intrinsically universal cellular automata are quite simple to construct in dimension two with few states: Banks does it with 2 states and von Neumann neighborhood with reflection and rotation symmetry.

5. Turing Universality

In dimension one, boolean circuit encoding is more puzzling as wire crossing capabilities is bounded by the local rule. Thus, historically, computation universality is achieved by direct simulation of universal models of computations :

Turing machines: Turing machines are easy to encode on cellular automata (see below) as an infinite tape really looks like a configuration of a cellular automaton.

In fact, several variants of Turing machines exist and an important literature on universality in the Turing world provide useful objects to build small universal automaton based on this model. The question of existence of small universal Turing machines was first raised by Shannon [73], different variants of Turing machines are discussed by Fischer [26]. For a survey on small Turing machine construction, see Woods and Neary [86].

Tag systems: Tag systems provide a better model to design very small universal objects. In fact, very small universal Turing machines are constructed by simulation of tag systems and their variants as originally proposed by Minsky [51, 13]. The original drawback of tag system was its exponential slow-down when simulating Turing machines, this drawback was removed recently by Woods and Neary [85, 86] achieving polynomial time simulation. The Turing-universality of rule 110 is obtained by Cook [15] by direct simulation of a proper variant of tag systems.

The variant of Turing machine we use is the following. A *Turing machine* is a tuple (S, Σ, B, s_0, T) where S is a finite set of states, Σ is a finite alphabet with a special blank symbol $B \in \Sigma$, $s_0 \in S$ is the initial state and $T : S \times \Sigma \to S \times \Sigma \times \{\leftarrow, \rightarrow\}$ is a partial transition map. A transition rule $T(s, a) = (s', b, d)$ reads as follow: when reading a from state s, write b on the tape, move in direction d, and enter state s'. A configuration of the machine is a triple (s, z, c) where $s \in S$ is the current state of the machine, $z \in \mathbb{Z}$ is the position of the head, and $c \in S^{\mathbb{Z}}$ is the content of the tape. The machines goes in one step from a configuration (s, z, c) to a configuration (s', z', c') if the transition rule $T(s, c(z)) = (s'', d, b)$ is defined and verifies $s' = s''$, $z' - z = d$, $c'(z) = b$ and for all position $z'' \neq z$, $c'(z) = c(z)$. Starting from a configuration $\mathfrak{c}$, an halting computation of the machine in time t consists of a sequence of configurations $(\mathfrak{c}_i)_{i=0}^{t}$ such that $\mathfrak{c}_0 = \mathfrak{c}$, the machine cannot reach any configuration from $\mathfrak{c}_t$ and for all i, the machine goes in one step from $\mathfrak{c}_i$ to $\mathfrak{c}_{i+1}$. The configuration $\mathfrak{c}_t$ is the output of the computation.

Following Smith III [74], a given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows. Let $S' = \Sigma \cup S \times \Sigma$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: $c'(z) = (s, c(z))$ and for all positions $z' \neq z$, $c'(z') = c(z)$. The local rule encodes the transition function of the Turing machine. For each transition $T(s, a) = (s', b, \leftarrow)$, for all states $x, y \in S$, $f(x, y, (s, a)) = (s', y)$ and $f(x, (s, a), y) = b$. Symmetrically, for each transition $T(s, a) = (s', b, \rightarrow)$, for all states $x, y \in S$, $f((s, a), y, x) = (s', y)$ and $f(x, (s, a), y) = b$. All undefined transitions apply identity: $f(x, y, z) = y$. With this encoding, starting from an encoded configuration $\tau(\mathfrak{c})$, the configuration evolves in one step to a configuration $\tau(\mathfrak{c}')$ where $\mathfrak{c}'$ is the next computation step of the Turing machine if it exists, $\mathfrak{c}' = \mathfrak{c}$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $(m + 1)n$ states.

To lower the number of states, Lindgren and Nordahl [41] introduce a simulation scheme where each step of the Turing machine computation is emulated by two time steps in the cellular automaton. A given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows. Let $S' = \Sigma \cup S \cup \{\bullet, \leftrightarrow\}$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: for all $z' < z$, $c'(2z') = \bullet$, $c'(2z' + 1) = c(z)$; for all $z' > z$, $c'(2z' + 1) = \bullet$, $c'(2z' + 2) = c(z)$; $c'(2z) = \bullet$ and either $c'(2z + 1) = s$, $c'(2z + 2) = c(z)$

or $c'(2z+1) = c(z)$, $c'(2z+2) = s$ (two possible encodings). The local rule encode the transition function of the Turing machine. Applying the rule: for each transition $T(s,a) = (s', b, \leftarrow)$, $f(\bullet, s, a) = s'$, $f(s, a, \bullet) = b$, $f(\bullet, a, s) = s'$, $f(a, s, \bullet) = b$, for each transition $T(s,a) = (s', b, \rightarrow)$, $f(\bullet, s, a) = b$, $f(s, a, \bullet) = s'$, $f(\bullet, a, s) = b$, $f(a, s, \bullet) = s'$. Moving: for all $s \in S$, $a, b \in \Sigma$, $f(\leftrightarrow, s, a) = \bullet$, $f(a, \leftrightarrow, s) = s$ $f(a, s, \leftrightarrow) = \bullet$, $f(s, \leftrightarrow, a) = s$. All undefined transitions apply the identity rule but for $\bullet$ and $\leftrightarrow$ that alternates: for all states $x, y \in S$, $f(x, \bullet, y) = \leftrightarrow$ and $f(x, \leftrightarrow, y) = \bullet$ With this encoding, starting from an encoded configuration $\tau(\mathfrak{c})$, the configuration evolves in two steps to a configuration $\tau(\mathfrak{c}')$ where $\mathfrak{c}'$ is the next computation step of the Turing machine if it exists, $\mathfrak{c}' = \mathfrak{c}$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $m + n + 2$ states.

Simulation of tag systems is more tricky due to the non-locality of one computation step of the system. Following Cook [15], one can consider cyclic tag systems. A cyclic tag system is given as a finite set of words $(w_0, \ldots, w_{N-1})$ on the alphabet $\{\circ, \bullet\}$. A configuration of the system is a word $u \in \{\circ, \bullet\}^*$. At time step t, the configuration u evolves to a configuraton v according if either $u_0 = \circ$ and $u_0 v = u$, either $u_0 = \bullet$ and $u_0 v = u w_{t \mod N}$. Cyclic tag systems can encode any recursive function. To encode all cyclic tag systems in a same cellular automaton $(1, S, \{-1, 0, 1\}, f)$, one can follow the following principle. Encode each configuration u of a cyclic tag system $(w_0, \ldots, w_{N-1})$ as a configuration of the kind $^\omega(T\_^k) \cdot u \cdot \blacksquare (w_0 \blacktriangle \ldots \blacktriangle w_{N-1})^\omega$ where intuitively T is a clock signal, $\blacksquare$ is the frontier between u and the rule and the rule is repeated on the right each word separated by a $\blacktriangle$. Giving the complete local rule is tedious but let us sketch its principle: each time a clock signal hits the first letter of u, it erases it and send a signal to the right transporting the value of that letter; when the signal meets the $\blacksquare$ it removes it and begins to treat the w word on the right, either erasing it or just crossing it; when the signal meets a $\blacktriangle$, it changes it into a $\blacksquare$ and the signal disappears. This principle is used by Cook to simulate cyclic tag systems with rule 110 particles and collisions.

A main point of discussion there is to decide which kind of configurations are acceptable for encoding Turing-universal computation. Finite configurations are certainly not a problem and using any, potentially non-recursive, configuration would permit trivial cellular automata to be misleadingly called universal. The previous constructions involving Turing machines use finite or ultimately periodic configurations with a same period on both sides, the same one for all simulated machines, whereas the tag system encoding uses ultimately periodic configurations with different periods, moreover these periodic parts depend on the simulated tag system. The tag system really needs this ultimate information, transforming the simulating cellular automaton into one working on finite configurations would have a constant but large impact on the number of states. As pointed in Durand and Róka [22], this configuration encoding problem adds difficulties to the formal definition of Turing-universality.

6. Intrinsic Universality

Even if the concept of intrinsically universal cellular automata took some time to emerge, intrinsic universality does not require more complex constructions to be achieved. Several technics are used to construct them:

Parallel Turing machines table lookup: A simple way to achieve intrinsic universality is to use synchronized parallel Turing heads (one copy of the same Turing machine per encoded cell) to lookup in the transition table (one copy in each encoded cell) of the encoded cellular automaton. Notice that the Turing machines used for this are not the same ones that are Turing-universal. In fact, their computational power is very small but they can carry precise information movement tasks.

One-way totalistic lookup: Another more cellular automata centric way to achieve intrinsic universality is, following Albert and Čulik 1987 [1], to simplify the task of the previous machine by simulating only one-way totalistic cellular automata which are sufficient to simulate all cellular automata.

Signals: The previous models are complex because the information displacement involved is still complex due to the sequential behavior of head-based machines. Following Ollinger [64] and Richard [71], particles and collisions, that is signals in the style of Mazoyer and Terrier [48], can be used to encode the information and perform the lookup task with parallel information displacement.

We explain here the parallel Turing machines table lookup technic, the other ones being refinements based on it. The one-dimensional first-neighbors universal cellular automaton $\mathcal{U}$ simulates a cellular automaton $(1, S, \{-1, 0, 1\}, f)$ the following way. Each configuration c is encoded as the concatenation of $\psi_{\mathcal{A}}(c(z))$ for all z. For each state $s \in S$, $\psi_{\mathcal{A}}(s)$ is a word of the kind $\blacksquare\tau(f(1,1,1)) \bullet \tau(f(1,1,2)) \bullet \ldots \bullet \tau(f(N,N,N))\blacktriangle 0^k\tau(s)0^k0^k$ where N is the size of S, k is the number of bits needed to encode numbers from 1 to N and $\tau(s)$ is a binary encoding of the state s. The simulation proceeds as follows so that the movement of each head is the same, up to translation, on each well encoded configuration. First the $\blacksquare$ letter is activated as a Turing head in initial state. The Turing head then moves to the left and copies the state $\tau(s_L)$ of the left neighbor in place of the first 0^k block. Then it symmetrically copies the state $\tau(s_R)$ of the right neighbor in place of the second 0^k block. This being done, the head scans the entire transition table, incrementing a counter (for example stored on top of the encoded states) at each step: if at some point the counter is equal to the triple of states, the result is copied from the transition table to the third 0^k block. At the end of the scan, the counter information is cleared, the result of the transition is copied in the $\tau(s)$ place and all three 0^k blocks are restored. The head then goes back to the $\blacksquare$. Using this simulation, one step of the simulated cellular automaton is simulated in a constant number of step for each cell by each Turing head. The universal automaton does not depend on the simulated automaton and is so intrinsically universal. A careful design can lead to less than 20 states. If the simulation uses the one-way totalistic technic, encoding states in unary, then it is easy to go under 10 states.

Notice that general Turing-universal cellular automata construction schemes from previous section concerning Turing machines can be adapted to produce small intrinsically universal cellular automata: apply the encoding schemes on machines performing the cell simulation task. However, the tag system simulations do not provide direct way to obtain intrinsic universality. Moreover, it is possible to design cellular automata Turing-universal by tag system simulation and not intrinsically universal.

More exotic intrinsically universal cellular automata have been studied on constrained rules. As an example, Moreira [54] constructed number conserving intrinsically universal automata, Bartlett and Garzon [6, 7] and Ollinger [62] do the same for bilinear cellular

automata, and Theyssier [79] for captive cellular automata for which he proves that almost all captive automata are intrinsically universal.

7. Reversibility and Universality

Reversible cellular automata are special in the sense that they can achieve Turing-universality as any Turing machine can be simulated by a reversible Turing machine but they cannot achieve intrinsic universality: reversible cellular automata only simulate reversible cellular automata. However, there exists reversible cellular automata which are universal with respect to the class of reversible cellular automata.

Definition 7.1 (reversible intrinsic universality)**.** A reversible cellular automaton $\mathcal{U}$ is intrinsically universal for reversible cellular automata if for each reversible cellular automaton $\mathcal{A}$ of the same dimension there exists an unpacking map o_m, a positive integer $n \in \mathbb{N}$ and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Turing-universality and weak form of intrinsic universality have been proposed by Morita [56], Morita and Imai [58, 59], Durand-Lose [23, 24], Miller and Fredkin [50]. As for classical cellular automata in higher dimension the simulation of reversible boolean circuits automatically gives reversible intrinsic universality.

For one-dimensional cellular automata, reversible intrinsic universality can be achieved by simulating any one-way reversible partitioned reversible cellular automaton with a first-neigbors reversible partitioned reversible cellular automaton. We briefly sketch how to use a scheme similar to parallel Turing machine table lookup with reversible Turing machines to achieve this goal. The first adaptation is to remark that a the local partition rule of a reversible automaton is a permutation, thus it can be encoded as a finite sequence of permutation pairs. So, the table of transition is encoded as a finite sequence of pairs of states. The reversible Turing machine task is to scan the transition table and for each pair it contains to replace the actual state by the second element of the pair if the state appears in the current pair. It is technical but straightforward to see that a reversible Turing machine can achieve this. The information movement is reversible as in partitioned automata each cell gives half its state to a neighbor and take half a state from the other without erasing any information. Developing this simulation scheme, one constructs a reversible intrinsically universal cellular automaton.

8. Conclusion

This brief reading guide has given the reader the keys both to further explore the literature and to construct by itself conceptually simple Turing-universal and/or intrinsically universal cellular automata in one and two dimensions. The following broad bibliography can be explored with the help of the main text, all references being cited.

References

1. J. Albert and K. Čulik, II, *A simple universal cellular automaton and its one-way and totalistic version*, Complex Systems **1** (1987), no. 1, 1–16.
2. S. Amoroso and Y. N. Patt, *Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures*, Journal of Computer and System Sciences **6** (1972), 448–464.
3. M. A. Arbib, *Simple self-reproducing universal automata*, Information and Control **9** (1966), no. 2, 177–189.
4. E. R. Banks, *Universality in cellular automata*, Symposium on Switching and Automata Theory (Santa Monica, California, 1970), IEEE, 1970, pp. 194–215.
5. ———, *Information processing and transmission in cellular automata*, Ph.D. thesis, Massachusetts Institute of Technology, 1971.
6. R. Bartlett and M. Garzon, *Monomial cellular automata*, Complex Systems **7** (1993), no. 5, 367–388.
7. ———, *Bilinear cellular automata*, Complex Systems **9** (1995), no. 6, 455–476.
8. C. H. Bennett, *Logical reversibility of computation*, IBM Journal of Research and Development **17** (1973), no. 6, 525–532.
9. E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning ways for your mathematical plays. Vol. 2*, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982, Games in particular.
10. A. W. Burks, *Programming and the theory of automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 65–83 (Essay Two).
11. ———, *Towards a theory of automata based on more realistic primitive elements*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 84–102 (Essay Three).
12. ———, *Von neumann's self-reproducing automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 3–64 (Essay One).
13. J. Cocke and M. Minsky, *Universality of tag systems with p=2*, J. ACM **11** (1964), no. 1, 15–20.
14. E. F. Codd, *Cellular automata*, Academic Press, New York, 1968.
15. M. Cook, *Universality in elementary cellular automata*, Complex Systems **15** (2004), 1–40.
16. M. D. Davis, *A note on universal turing machines*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 167–175.
17. ———, *The universal computer: The road from leibniz to turing*, W. W. Norton & Company, 2000.
18. M. Delorme, *An introduction to cellular automata: some basic definitions and concepts*, Cellular automata (Saissac, 1996) (M. Delorme and J. Mazoyer, eds.), Kluwer Acad. Publ., Dordrecht, 1999, pp. 5–49.
19. J.C. Delvenne, P. Kurka, and V. D. Blondel, *Decidability and universality in symbolic dynamical systems*, Fundam. Inform. **74** (2006), no. 4, 463–490.
20. A. K. Dewdney, *The cellular automata programs that create wireworld, rugworld and other diversions*, Scientific American **262** (1990), 146–149.
21. Jean-Christophe Dubacq, *How to simulate turing machines by invertible one-dimensional cellular automata*, Int. J. Found. Comput. Sci. **6** (1995), no. 4, 395–402.
22. B. Durand and Zs. Róka, *The game of life: universality revisited*, Cellular automata (Saissac, 1996) (M. Delorme and J. Mazoyer, eds.), Kluwer Acad. Publ., Dordrecht, 1999, pp. 51–74.
23. J. Durand-Lose, *Reversible cellular automaton able to simulate any other reversible one using partitioning automata*, LATIN '95 (R. A. Baeza-Yates, E. Goles Ch., and P. V. Poblete, eds.), Lecture Notes in Computer Science, vol. 911, Springer, 1995, pp. 230–244.
24. ———, *Automates cellulaires, automates partitions et tas de sable*, Ph.D. thesis, Université Bordeaux I, 1996.
25. ———, *Intrinsic universality of a 1-dimensional reversible cellular automaton*, STACS 97 (Lübeck), Lecture Notes in Comput. Sci., vol. 1200, Springer, Berlin, 1997, pp. 439–450.
26. Patrick C. Fischer, *On formalisms for turing machines*, J. ACM **12** (1965), no. 4, 570–580.
27. E. Fredkin and T. Toffoli, *Conservative logic*, International Journal of Theoretical Physics **21** (1982), 219–253.
28. A. Gajardo and E. Goles Ch., *Universal cellular automaton over a hexagonal tiling with 3 states*, IJAC **11** (2001), no. 3, 335–354.
29. M. Gardner, *Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'*, Scientific American **223** (1970), no. 4, 120–123.

30. K. Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, Monatschefte für Mathematik und Physik **38** (1931), 173–198, reprinted in S. Feferman, W. Dawson, S.C. Kleene, G. Moore, R.M. Solovay, J. van Heijendort (eds.), *Kurt Gödel: Collected Works*, Oxford University Press, Oxford, **1** (1986), 144–195.
31. G. A. Hedlund, *Endormorphisms and automorphisms of the shift dynamical system*, Mathematical Systems Theory **3** (1969), 320–375.
32. Hertling, *Embedding cellular automata into reversible ones*, Unconventional Models of Computation, Springer, 1998 (C. S. Calude, J. Casti, and M. J. Dinneen, eds.), 1998.
33. K. Imai and K. Morita, *A computation-universal two-dimensional 8-state triangular reversible cellular automaton*, Theoretical Computer Science **231** (2000), no. 2, 181–191.
34. J. Kari, *Reversibility of 2D cellular automata is undecidable*, Physica D. Nonlinear Phenomena **45** (1990), no. 1-3, 379–385, Cellular automata: theory and experiment (Los Alamos, NM, 1989).
35. ———, *Reversibility and surjectivity problems of cellular automata*, J. Comput. Syst. Sci. **48** (1994), no. 1, 149–182.
36. ———, *Theory of cellular automata: A survey*, Theoretical Computer Science **334** (2005), 3–33.
37. S. C. Kleene, *Representation of events in nerve nets and finite automata*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 3–41.
38. G. Lafitte, *Gödel incompleteness revisited*, personal communication *(submitted to JAC 2008)*, 2008.
39. C.G. Langton, *Self-reproduction in cellular automata*, Physica D **10** (1984), no. 1-2, 135–144.
40. Y. Lecerf, *Machines de turing réversibles*, C. R. Acad. Sci. Paris **257** (1963), 2597–2600.
41. K. Lindgren and M. G. Nordahl, *Universal computation in simple one-dimensional cellular automata*, Complex Systems **4** (1990), no. 3, 299–318.
42. N. Margolus, *Physics-like models of computation*, Physica D **10** (1984), 81–95.
43. B. Martin, *Construction modulaire d'automates cellulaires*, Ph.D. thesis, École Normale Supérieure de Lyon, 1993.
44. B. Martin, *A universal cellular automaton in quasilinear time and its S-m-n form*, Theoretical Computer Science **123** (1994), no. 2, 199–237.
45. J. Mazoyer, *Computations on one dimensional cellular automata*, Ann. Math. Artif. Intell. **16** (1996), 285–309.
46. ———, *Computations on grids*, Cellular automata (Saissac, 1996), Kluwer Acad. Publ., Dordrecht, 1999, pp. 119–149.
47. J. Mazoyer and Ivan Rapaport, *Inducing an order on cellular automata by a grouping operation*, Discrete Applied Mathematics **91** (1999), no. 1-3, 177–196.
48. J. Mazoyer and Véronique Terrier, *Signals in one-dimensional cellular automata*, Theoretical Computer Science **217** (1999), no. 1, 53–80, Cellular automata (Milan, 1996).
49. W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics **5** (1943), 115–133.
50. D. B. Miller and E. Fredkin, *Two-state, reversible, universal cellular automata in three dimensions*, Conf. Computing Frontiers (N. Bagherzadeh, M. Valero, and A. Ramírez, eds.), ACM, 2005, pp. 45–51.
51. M. Minsky, *Computation: Finite and infinite machines*, Prentice Hall, Englewoods Cliffs, 1967.
52. E. F. Moore, *Machine models of self-reproduction*, Proceedings of Symposia in Applied Mathematics, vol. 14, American Mathematical Society, 1962, pp. 17–33.
53. ———, *Machine models of self-reproduction*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 187–203 (Essay Six).
54. A. Moreira, *Universality and decidability of number-conserving cellular automata*, Theoretical Computer Science **292** (2003), no. 3, 711–721.
55. K. Morita, *A simple construction method of a reversible finite automaton out of fredkin gates, and its related problem*, IEICE Trans. Inf. & Syst. **E73** (1990), no. 6, 978–984.
56. ———, *Reversible simulation of one-dimensional irreversible cellular automata*, Theoretical Computer Science **148** (1995), no. 1, 157–163.
57. K. Morita and M. Harao, *Computation universality of one-dimensional reversible (injective) cellular automata*, IEICE Trans. Inf. & Syst. **E72** (1989), 758–762.
58. K. Morita and K. Imai, *Self-reproduction in a reversible cellular space*, Theoretical Computer Science **168** (1996), no. 2, 337–366.

59. ———, *Number-conserving reversible cellular automata and their computation-universality*, Theoretical Informatics and Applications **35** (2001), no. 3, 239–258.
60. T. Neary and D. Woods, *P-completeness of cellular automaton rule 110*, Proceedings of ICALP 2006, Lecture Notes in Computer Science, vol. 4051, Springer, Berlin, 2006, pp. 132–143.
61. F. Noural and R.S. Kashef, *A universal four-state cellular computer*, IEEE Transactions on Computers **24** (1975), no. 8, 766–776.
62. N. Ollinger, *Two-states bilinear intrinsically universal cellular automata*, Fundamentals of computation theory (Riga, Latvia, 2001) (Berlin) (R. Freivalds, ed.), Lecture Notes in Computer Science, vol. 2138, Springer, 2001, pp. 396–399.
63. ———, *Automates cellulaires : structures*, Ph.D. thesis, École Normale Supérieure de Lyon, 2002.
64. ———, *The quest for small universal cellular automata*, International Colloquium on Automata, languages and programming (Málaga, Spain, 2002) (Berlin) (P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, eds.), Lecture Notes in Computer Science, vol. 2380, Springer, 2002, pp. 318–329.
65. ———, *The intrinsic universality problem of one-dimensional cellular automata*, Symposium on Theoretical Aspects of Computer Science (Berlin, Germany, 2003), Lecture Notes in Computer Science, Springer, Berlin, 2003, (to appear).
66. D. Perrin, *Les débuts de la théorie des automates*, Technique et Science Informatique **14** (1995), 409–433.
67. E. Post, *The two-valued iterative systems of mathematical logic*, Princeton University Press, Princeton, 1941.
68. ———, *Formal reductions of the general combinatorial decision problem*, American Journal of Mathematics **65** (1943), no. 2, 197–215.
69. I. Rapaport, *Inducing an order on cellular automata by a grouping operation*, Ph.D. thesis, École Normale Supérieure de Lyon, 1998.
70. G. Richard, *From turing machine to rule 110: embedding computational power in small cellular automata through particles and collisions*, personal communication *(submitted to JAC 2008)*, 2008.
71. ———, *A particular universal cellular automaton*, personal communication, 2008.
72. D. Richardson, *Tessellations with local transformations*, Journal of Computer and System Sciences **6** (1972), 373–388.
73. C. E. Shannon, *A universal turing machine with two internal states*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 157–165.
74. A. R. Smith, III, *Simple computation-universal cellular spaces*, Journal of the ACM **18** (1971), 339–353.
75. K. Sutner, *Universality and cellular automata*, MCU 2004 (M. Margenstern, ed.), Lecture Notes in Computer Science, vol. 3354, Springer, 2004, pp. 50–59.
76. J. W. Thatcher, *Self-describing turing machines and self-reproducing cellular automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 103–131 (Essay Four).
77. ———, *Universality in the von neumann cellular model*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 132–186 (Essay Five).
78. G. Theyssier, *Automates cellulaires : un modèle de complexités*, Ph.D. thesis, École Normale Supérieure de Lyon, 2005.
79. ———, *How common can be universality for cellular automata?*, STACS 2005, Lecture Notes in Computer Science, vol. 3404, Springer, 2005, pp. 121–132.
80. Tommaso Toffoli, *Computation and construction universality of reversible cellular automata*, J. Comput. Syst. Sci. **15** (1977), no. 2, 213–231.
81. A. M. Turing, *On computable numbers with an application to the entscheidungs problem*, Proceedings of the London Mathematical Society 2 **42** (1936), 230–265.
82. J. von Neumann, *Theory of self-reproducing automata*, University of Illinois Press, Urbana, Ill., 1966, (A. W. Burks, ed.).
83. S. Wolfram, *Universality and complexity in cellular automata*, Physica D. Nonlinear Phenomena **10** (1984), no. 1-2, 1–35, Cellular automata (Los Alamos, N.M., 1983).
84. S. Wolfram, *A new kind of science*, Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.
85. D. Woods and T. Neary, *On the time complexity of 2-tag systems and small universal turing machines*, FOCS 2006, IEEE Computer Society, 2006, pp. 439–448.
86. ———, *The complexity of small universal turing machines*, Computability in Europe (S. B. Cooper, B. Löwe, and A. Sorbi, eds.), Lecture Notes in Computer Science, vol. 4497, Springer, 2007, pp. 791–799.